Week 13: Activity Key

Last Week

- Spatial EDA
- GP models to spatial data
- Spatial Prediction / Model Choice
- Anisotropic Spatial Models

This Week

- GLM models
- Spatial GLMs

Generalized Linear Model Notation

There are three components to a generalized linear model:

1. Sampling Distribution: such as Poisson or Binomial

2. Linear combination of predictors: $\eta = X\beta$

3. A link function to map the linear combination of predictors to the support of the sampling distribution.

Binary Regression Overview

Write out the complete model specification for binary regression.

- Assume Y_i is the binary response for the i^{th} observation,

$$\begin{aligned} Y_i &\sim Bernoulli(\pi_i) \\ logit(\pi_i) &= X_i\beta, \\ \text{or } \Phi^{-1}(\pi_i) &= X_i\beta \end{aligned}$$

- where $logit(\pi_i) = log\left(\frac{\pi_i}{1-\pi_i}\right)$

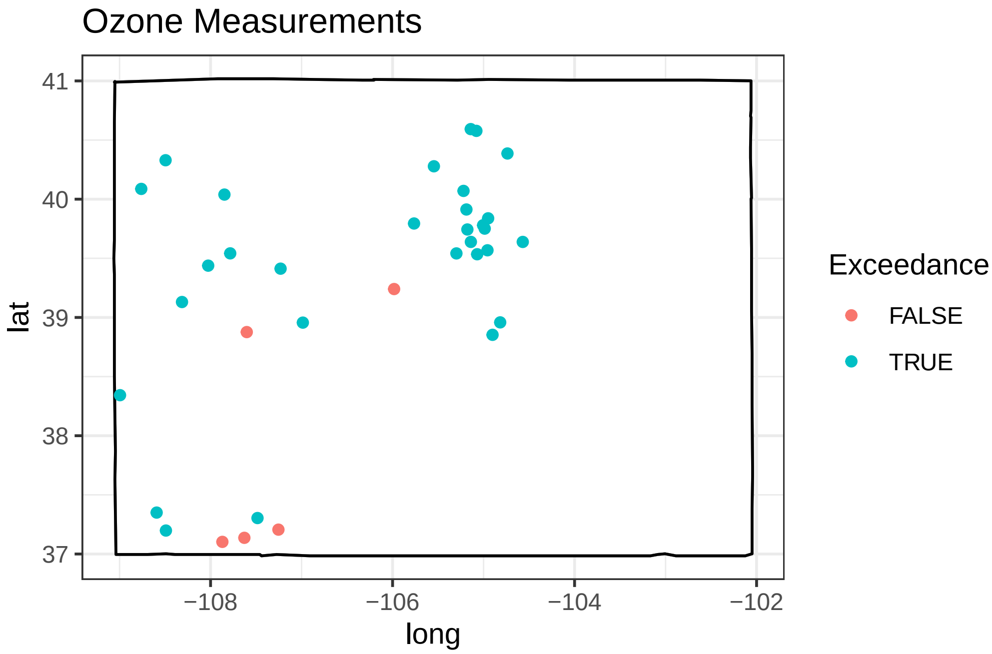
- where $\Phi() =$ is the CDF of a standard normal distribution

Latent interpretation of probit model:

Let $z_i > 0$ if $y_i = 1$. Otherwise, let $z_i < 0$. Then $z_i \sim N(X\beta, 1)$ is a latent continuous variable that is mapped to zero or 1.

For a set of predictors, X', then $z' \sim N(X'\beta, 1)$ and the probability of a 1 or zero can be obtained by integrating the latent distribution.

Consider air quality data from Colorado as a motivating example.



Interpret the output.

```r
CO <- CO %>% mutate(north = as.numeric(Latitude > 38 ))
glm(Exceedance~north, family=binomial(link = 'probit'),data=CO) %>% display()
```

```
glm(formula = Exceedance ~ north, family = binomial(link = "probit"), 
    data = CO)
            coef.est coef.se
(Intercept) 0.00     0.51   
north       1.48     0.62   
---
  n = 35, k = 2
  residual deviance = 22.9, null deviance = 28.7 (difference = 5.8)
```

```r
glm(Exceedance~north, family=binomial(link = 'logit'),data=CO) %>% display()
```

```
glm(formula = Exceedance ~ north, family = binomial(link = "logit"), 
    data = CO)
            coef.est coef.se
(Intercept) 0.00     0.82   
north       2.60     1.10   
---
  n = 35, k = 2
  residual deviance = 22.9, null deviance = 28.7 (difference = 5.8)
```

Spatial Binary Regression

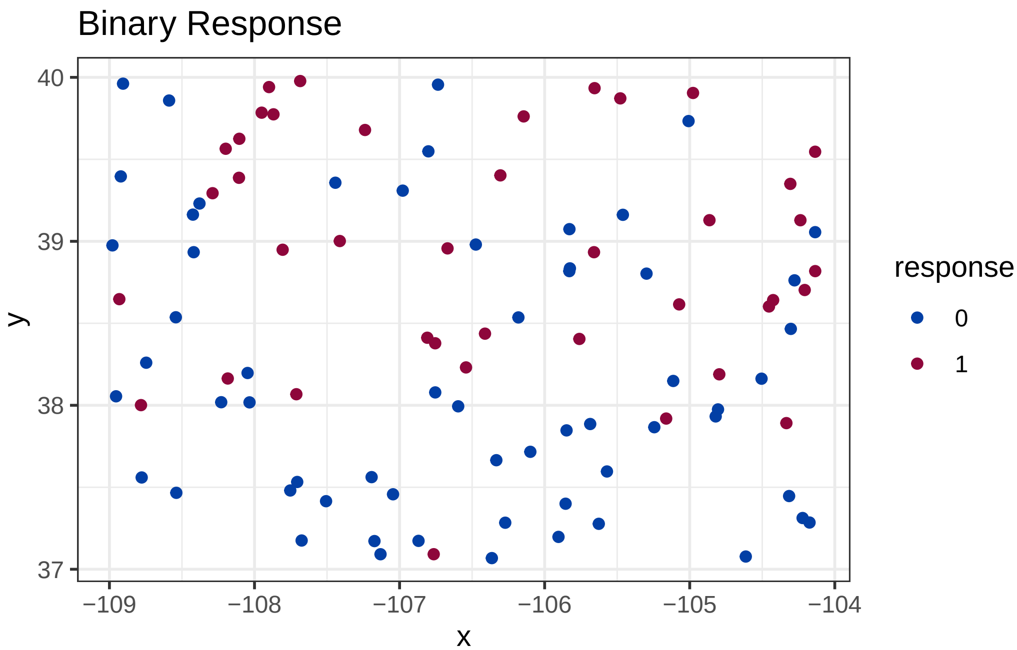
Assume $Y(s_i)$ is the binary response for s_i,

$$\begin{aligned} Y(s_i)|\beta, w(s_i) &\sim Bernoulli(\pi(s_i)) \\ \Phi^{-1}(\pi(s_i)) &= X(s_i)\beta + w(s_i), \end{aligned}$$

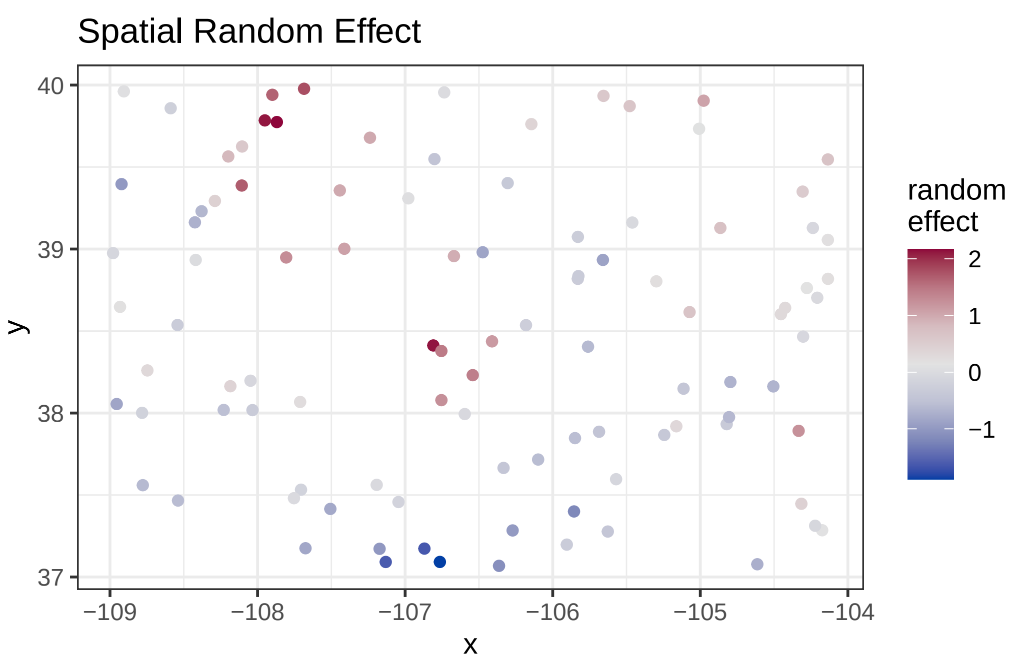
where $W \sim N(0, \sigma^2 H(\phi))$

Simulating spatial random effects for binary data

```r
N.sim <- 100
Lat.sim <- runif(N.sim,37,40)
Long.sim <- runif(N.sim,-109,-104)
phi.sim <- 1
sigmasq.sim <- 1
beta.sim <- c(-1,1)
north.sim <-  as.numeric(Lat.sim > 38)


d <- dist(cbind(Lat.sim,Long.sim), upper = T, diag = T) %>% as.matrix
H.sim <- sigmasq.sim * exp(- d / phi.sim)
w.sim <- rmnorm(1,0,H.sim)
xb.sim <- beta.sim[1] + beta.sim[2] * north.sim
y.sim <- rbinom(N.sim,1,pnorm(xb.sim + w.sim))
```



Binary Response
40
39
38
37
y
−109
−108
−107
−106
−105
−104
x
response
0
1



Spatial Random Effect
40
39
38
37
y
−109
−108
−107
−106
−105
−104
x
random effect
2
1
0
−1

STAN: probit regression

```
// The input data is a vector 'y' of length 'N'.
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
  vector[N] x;
}

// The parameters accepted by the model. Our model
// accepts two parameters 'mu' and 'sigma'.
parameters {
  real beta0;
  real beta1;
}

transformed parameters {
  real<lower = 0, upper = 1> p[N];
  for (i in 1:N) {
    p[i] = Phi(beta0 + beta1 * x[i]);
  }
}


// The model to be estimated.
model {
  for (i in 1:N){
    y[i] ~ bernoulli(p[i]);
  }
}
```

Binary Regression

```r
probit_stan <- stan(file = 'probit_regression.stan', data = list(N = N.sim, y = y.sim, x = no
```

```r
print(probit_stan, pars = c('beta0', 'beta1'))
```

```
Inference for Stan model: anon_model.
```

```
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
beta0 -1.35    0.01 0.32 -2.05 -1.55 -1.34 -1.14 -0.78   727    1
beta1  1.47    0.01 0.35  0.82  1.23  1.45  1.68  2.22   736    1

Samples were drawn using NUTS(diag_e) at Mon Apr  7 22:31:13 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

```
glm(y.sim ~ north.sim, family = binomial(link = 'probit'))
```

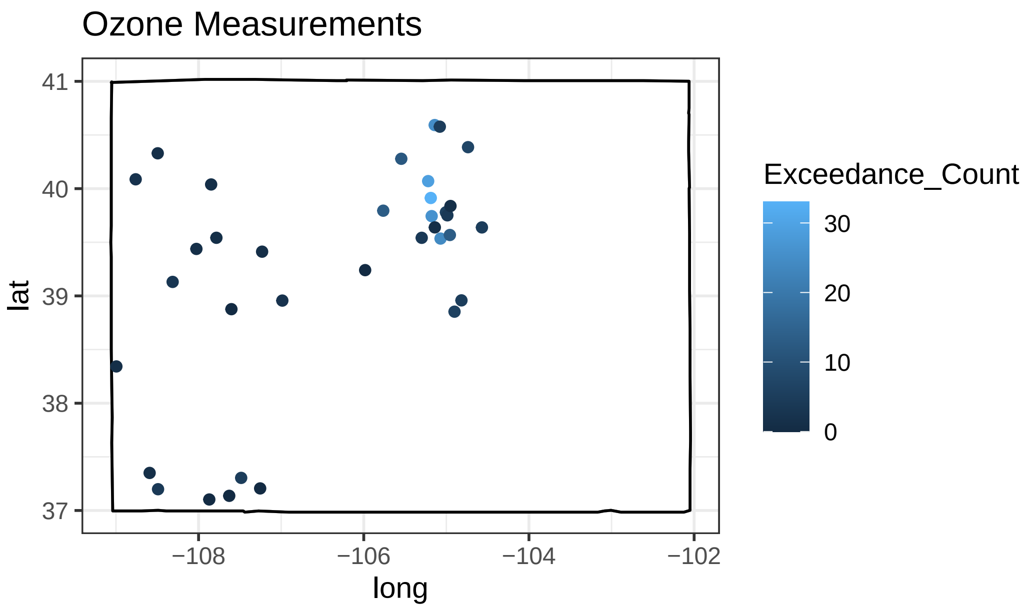
```
Call:  glm(formula = y.sim ~ north.sim, family = binomial(link = "probit"))

Coefficients:
(Intercept)    north.sim
     -1.318        1.429

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      134.6
Residual Deviance: 113.7    AIC: 117.7
```

```
tibble(y.sim = y.sim, north.sim = north.sim) %>% stan_glm(y.sim ~ north.sim, family = binomia
```

```
stan_glm
 family:       binomial [probit]
 formula:      y.sim ~ north.sim
 observations: 100
 predictors:   2
------
            Median MAD_SD
(Intercept) -1.3    0.3
north.sim    1.4    0.3

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Spatial Poisson Regression

Motivation



Poisson Regression Overview

Write out the complete model specification for Poisson regression.

Assume Y_i is the count response for the i^{th} observation,

$$\begin{aligned} Y_i &\sim Poisson(\lambda_i) \\ \log(\lambda_i) &= X_i\beta, \end{aligned}$$

thus $\exp(X_i\beta) \geq 0$

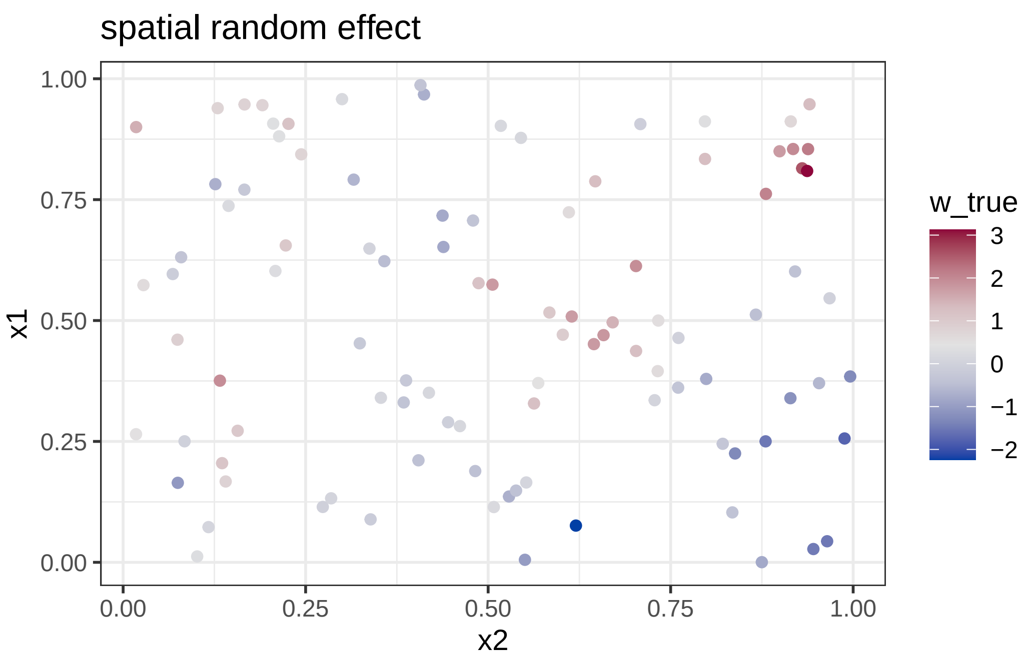
Next write out a Poisson regression model with spatial random effects

$$\begin{aligned} Y(s_i) &\sim Poisson(\lambda(s_i)) \\ \log(\lambda(s_i)) &= X(s_i)\beta + w(s_i), \end{aligned}$$

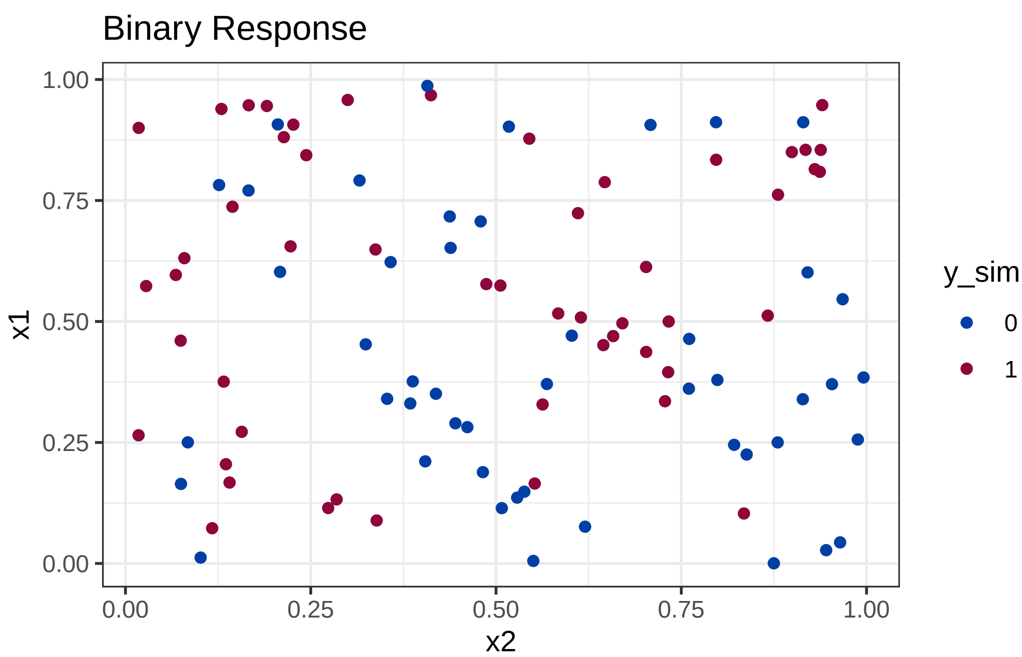
where $W \sim N(0, \sigma^2 H(\phi))$

1. Simulate and visualize spatial random effects for binary data: No Covariates

```
N <- 100
x1 <- runif(N)
x2 <- runif(N)
phi_true <- .2
sigmasq_true <- 1

d <- dist(cbind(x1,x2), upper = T, diag = T) %>% as.matrix
H <- sigmasq_true * exp(- d / phi_true)
w_true <- rmnorm(1,0,H)
p_true <- pnorm(w_true)
y_sim <- rbinom(N,1,p_true)

sim1_dat <- tibble(x1 = x1, x2 = x2, w_true = w_true, p_true = p_true, y_sim = as.factor(y_si

sim1_dat %>% ggplot(aes(y = x1, x = x2, color = w_true)) +
  geom_point() + theme_bw() +
  scale_color_gradientn(colours = colorspace::diverge_hcl(7)) +
  ggtitle('spatial random effect')
```



```r
sim1_dat %>% ggplot(aes(y = x1, x = x2, color = y_sim)) +
  geom_point() + theme_bw() +
  ggtitle('Binary Response') + scale_color_manual(values=c("#023FA5", "#8E063B"))
```



2. Fit a model for this setting

```stan
//
// This Stan program defines a simple model, with a
// vector of values 'y' modeled as normally distributed
// with mean 'mu' and standard deviation 'sigma'.
//
// Learn more about model development with Stan at:
//
//    http://mc-stan.org/users/interfaces/rstan.html
//    https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
//

// The input data is a vector 'y' of length 'N'.
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
```

```
  matrix[N,N] d;
}

// The parameters accepted by the model. Our model
// accepts two parameters 'mu' and 'sigma'.
parameters {
  vector[N] w;
  real <lower = 0.1, upper = .8> phi;
  real<lower = 0> sigmasq;
}

transformed parameters {
  real<lower = 0, upper = 1> p[N];
  vector[N] mu;
  corr_matrix[N] Sigma;

  for (i in 1:N) {
    p[i] = Phi(w[i]);
    mu[i] = 0;
  }
  for(i in 1:(N-1)){
   for(j in (i+1):N){
     Sigma[i,j] = exp((-1)*d[i,j]/ phi);
     Sigma[j,i] = Sigma[i,j];
   }
 }
 for(i in 1:N) Sigma[i,i] = 1;
}

// The model to be estimated.
model {
  w ~ multi_normal(mu, sigmasq * Sigma);
  for (i in 1:N){
    y[i] ~ bernoulli(p[i]);
  }
  sigmasq ~ inv_gamma(5,5);
}

Inference for Stan model: anon_model.
2 chains, each with iter=10000; warmup=5000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=10000.
```

```
        mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
phi     0.33    0.01 0.19 0.11 0.17 0.27 0.45  0.75   556 1.01
sigmasq 1.34    0.02 0.66 0.56 0.90 1.19 1.59  3.07   719 1.00

Samples were drawn using NUTS(diag_e) at Mon Apr  7 22:43:40 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```